A Synergy Coalition Group based Dynamic Programming Algorithm for Coalition Formation

Luke Riley
Dept. of Computer Science
University of Liverpool, UK
L.J.Riley@Liverpool.ac.uk

Katie Atkinson
Dept. of Computer Science
University of Liverpool, UK
K.M.Atkinson@Liverpool.ac.uk

Paul E. Dunne
Dept. of Computer Science
University of Liverpool, UK
P.E.Dunne@Liverpool.ac.uk

Terry R. Payne
Dept. of Computer Science
University of Liverpool, UK
T.R.Payne@Liverpool.ac.uk

ABSTRACT

Coalition formation in characteristic function games entails agents partitioning themselves into a coalition structure and assigning the numeric rewards of each coalition via a payoff vector. Various coalition structure generation algorithms have been proposed that guarantee that an optimal coalition structure is found. We present the Synergy Coalition Group-based Dynamic Programming (SCG-DP) algorithm that guarantees that an optimal coalition structure *and* a least core stable payoff vector is found. This is completed by extending the existing results for the Synergy Coalition Group (SCG) representation to show that only coalitions in the SCG are needed to find a weak-least core stable payoff vector. The SCG-DP algorithm builds on this result by performing only the search operations necessary to guarantee that coalitions in the SCG of the given characteristic function game are found. The number of operations required is significantly less for many coalition-value distributions compared to the original Dynamic Programming (DP) algorithm [34] that finds an optimal coalition structure (e.g. only $\sim$60% of DP's coalition lookup operations are performed in SCG-DP for 18 agents using a normal coalition-value distribution). Our experimental results show that a lower bound for these operations in SCG-DP converges onto 50%. This is an increase on the $\sim$33% bound of the optimal dynamic programming (ODP) algorithm [14], but ODP does not search for a stable solution.

General Terms

Algorithms, Economics, Theory

Keywords

Coalition Formation; Dynamic Programming; Synergy Coalition Groups; Cooperative Game Theory

1. INTRODUCTION

Coalition formation is a process whereby agents recognise that co-operation with others can occur in a mutually beneficial manner and therefore the agents can choose appropriate temporary groups (named coalitions) to form. But determining the set of coalitions that should form is difficult due to: (a) the exponential number of different possible coalitions; (b) the exponential number of possible sets of coalitions; and (c) the many ways in which the agents of a coalition can agree to distribute its rewards between its members (if this reward is transferable between the agents).

Therefore, coalition formation can be thought of as a three stage process [25, 27]: (1) calculating the value of each possible coalition; (2) finding an optimal set of coalitions; and (3) dividing the value of the coalitions in a manner that no agent can object to.

A set of coalitions, known as a *coalition structure*, is an *optimal* coalition structure if it has the - possibly joint - highest total value of all the possible coalition structures (i.e. it maximises social welfare). In *characteristic function games*, where the set of agents must be divided into exhaustive and disjoint coalitions, finding an optimal coalition structure is known as the *coalition structure generation problem* [27]. This problem has been proved to be NP-complete [27], yet there have been algorithms developed to combat this complexity [21]. The algorithms that provide a guarantee that an optimal coalition structure will be found, can be divided into the following two categories (for a more detailed description of these two categories see [24]):

1. **Dynamic programming algorithms:** This is the collection of algorithms where the main problem is solved by dividing it up into sub-problems and then solving the smallest first. The answers to the smallest problems help with answering the larger problems [8]. There are coalition structure generation dynamic programming algorithms (e.g. see [14, 19, 22]) that are modifications of an algorithm presented in [34], which solves the complete set partitioning problem. The main advantage of the dynamic programming algorithms is that an optimal coalition structure is guaranteed to be found in $O(3^n)$ time. Yet a disadvantage is that an optimal coalition structure is only returned when the algorithm completes.
2. **Tree search algorithms:** These algorithms divide the search space into subspaces that are searched using depth-first and branch-and-bound techniques, where the quality of the solution gets better over time (in some well behaved manner). Some example algorithms in this category are: [7, 15, 25, 27]. The main disadvantage of these algorithms is that the worst case running time is $O(n^n)$.

Additionally, there exists algorithms that combine aspects of both of these categories (e.g. see [14, 29]). However all of these coali-

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

tion structure generation algorithms do not consider the third stage of coalition formation; payoff distribution. When agents are adversarial, an agent will only join a coalition if it is given a payoff that is stable (i.e. a payoff that cannot reasonably be objected to[1]). It can even be beneficial for benevolent agents to only join a coalition if the payoff they receive is stable [4].

In this paper, the *Synergy Coalition Group-based Dynamic Programming* (SCG-DP) algorithm is introduced, that locates an optimal coalition structure and concurrently identifies the *Synergy Coalition Group* (SCG), which is used to guarantee that a weak-least core or weak-least core$^+$ stable payoff distribution is also found (where the plus sign indicates cross coalitional payoff transfers are allowed). The SCG-DP algorithm achieves these results, even though it cuts down significantly on the number of operations required by the original Dynamic Programming (DP) algorithm [34] for many different coalition-value distributions.

The rest of the paper is structured as follows: Section 2 introduces the background to characteristic function games, core-based stability concepts and the SCG. Section 3 introduces the theoretical foundations that SCG-DP is built upon, including a theorem that shows a weak-least core solution can always be found using the SCG. Section 4 details SCG-DP, discusses why SCG-DP satisfies certain solution concepts, and describes a SCG-DP example. Section 5 evaluates SCG-DP against the Optimal Dynamic Programming algorithm [14] and the DP algorithm [34], according to the number of split operations performed and the number of coalition lookup operations. Finally Section 6 concludes and describes future work.

2. BACKGROUND

2.1 Characteristic Function Games

The *characteristic function game* model of coalition formation [18] is denoted: $\mathcal{G} = \langle N, v\rangle$ where $N = \{1, 2, ..., n\}$ is the set of agents, and v is the *characteristic function* ($v(2^N) \rightarrow \mathbb{R}$) which assigns every possible coalition a real numeric payoff. Characteristic function games assumes that each coalition's value is static, deterministic and independent of the other coalitions that could form.

An *outcome* of a characteristic function game $\mathcal{G} = \langle N, v\rangle$ for n agents is a coalition structure and payoff vector pair, denoted: $\langle CS, x\rangle$, where CS is a set of k coalitions $CS = \{C_1, ..., C_k\}$ and x is the payoff vector $x = (x_1, ..., x_n)$. Occasionally in this paper, notation is abused so that $x(C) = \sum_{i\in C} x_i$. The coalition structure CS has the following properties: $\bigcup_{j=1}^{k} C_j = N$; and $C_i \cap C_j = \emptyset$ for any $i, j \in \{1, ..., k\}$ where $i \neq j$. To find a *stable outcome* of a characteristic function game, all the agents should have *no valid objection*. Stability is a necessary but not sufficient condition for a payoff vector (or coalition structure) to be agreed upon, because there can be multiple possible stable outcomes.

2.2 Core-based Stability Solution Concepts

The idea of a valid objection changes depending on which stability solution concept is used. Perhaps the most intuitive stability solution concept is known as the **core** [10]. The core is the set of stable outcomes where no subset of agents have an incentive to deviate, i.e., there is no payoff distribution that would make at least one member of a deviating coalition better off, without negatively affecting the other members of the deviating coalition[2].

[1]Definitions of stability are discussed in Section 2.2.

[2]In this section, the core is introduced, not the CS-core, as the CS-core has been shown to not be able to guarantee that the optimal coalition structure is CS-core stable [2, 28].

Definition 1: *The* **core***:- For a characteristic function game $\mathcal{G} = \langle N, v\rangle$, a payoff vector $x = (x_1, ..., x_n)$ for the grand coalition is in the core iff:*

$$\sum_{i\in N} x_i = v(N) \qquad \sum_{i\in C} x_i \geq v(C), \forall C \subset N$$

A problem with the core is that it can sometimes be empty. To tackle this issue, the concept of the ϵ*-core* was introduced in [30], where the ϵ-core is a more general case of the core. Two different definitions of the ϵ*-core* were introduced, named the *strong* ϵ*-core* and the *weak* ϵ*-core*. This paper focuses on the weak ϵ-core:

Definition 2: *The* **weak ϵ-core***:- For a characteristic function game $\mathcal{G} = \langle N, v\rangle$ and a value ϵ, a payoff vector $x = (x_1, ..., x_n)$ for the grand coalition is in the weak ϵ-core iff:*

$$\sum_{i\in N} x_i = v(N) \qquad \sum_{i\in C} x_i \geq v(C) - |C|\epsilon, \ \forall C \subset N$$

Notice when $\epsilon = 0$, the weak ϵ-core definition is equivalent to the core. The difference between the strong and weak ϵ-cores is that under the weak definition, the penalty of forming a new coalition is dependent on its size, while the penalty for forming a new coalition under the strong definition is a fixed amount for any coalition.

The *weak least core* is the smallest, non-empty weak ϵ-core [30]:

Definition 3: **weak least core***:- For a characteristic function game $\mathcal{G} = \langle N, v\rangle$, a payoff vector x is in the weak least core iff:*

x is in the weak ϵ-core and $\forall \epsilon' < \epsilon$, the weak ϵ'-core is empty

It should be mentioned that core/ϵ-core stable payoff vectors can be found with payoff vector transfer schemes (such as [3, 13, 33]), which take a characteristic function game outcome, iteratively apply an operation on it and converge to an element of the ϵ-core. Yet all of [3, 13, 33] do not detail the search for the optimal coalition structure (they assume the grand coalition forms) and only [13] presents a method to find an element of the least core.

Additionally, a non-cooperative game theoretic approach to coalition formation is through coalitional bargaining, which describes how stable characteristic function game outcomes occur via a process of non-cooperative bargaining and outlines equilibrium concepts that characterise the agent's behaviour in these games.

The coalitional bargaining papers of [5, 9, 17] detail how non-cooperative game theoretic solution concepts match the cooperative solution concept of the core. But these studies do not detail what *cooperative* game solution concept is found when the core is empty.

2.3 Synergy Coalition Groups

When searching for stability, it may not be necessary to check all possible coalitions. The synergy coalition group (SCG) [6] has been shown to include all the coalitions needed to guarantee that a core stable solution can be found. A coalition C is said to introduce synergy, if C has a greater value than any possible disjoint and non-overlapping partition of C. A SCG is defined as follows:

Definition 4: *A* ***synergy coalition group*** *W consists of a set of pairs of the form $(C, v(C))$. A pair is only in W if it details a coalition that gives synergy to its members. The singleton coalitions are assumed to always be represented within W. For any coalition $C \in (C, v(C)) \in W$, the value of C is $v(C)$. For any coalition $C' \notin W$, the value of C' is: $v(C') = \max(\sum_{C_j \in P_{C'}} v(C_j))$ where $P_{C'} = \{C_1, C_2, ..., C_k\}$ is a partition of C' such that: (a) $\bigcup_{i=1}^{k} C_i = C'$; (b) $C_i \cap C_j = \emptyset$ for any $C_i, C_j \in P_{C'}$ where $i \neq j$; and (c) $(C_j, v(C_j)) \in W$, for all $C_j \in P_{C'}$.*

If a coalition C' is not represented within W, then its value comes from its highest valued partition, as long as this partition has the following properties: (a) all the agents of C' are in a coalition of the partition and there are no additional agents in the partition; (b) each agent of C' occurs in only one coalition of the partition; and (c) each coalition in the partition is represented within W.

Example 1: *Consider the characteristic function game $G = \langle N, v\rangle$ where $N = \{1, 2, 3, 4\}$ and the SCG representation W gives the following valuations: $W = \{(\{1\}, 2), (\{2\}, 1), (\{3\}, 1), (\{4\}, 2), (\{1, 2\}, 5)\}$ (i.e. the utility value of agent 1 by itself is 2, while the utility value of the coalition, of agents 1 and 2, is 5)* .

Example values of coalitions that are not explicitly represented are: $v(\{3, 4\}) = v(\{3\}) + v(\{4\}) = 1 + 2 = 3$; $v(\{1, 2, 3\}) = v(\{1, 2\}) + v(\{3\}) = 5 + 1 = 6$; and $v(\{1, 2, 3, 4\}) = v(\{1, 2\}) + v(\{3\}) + v(\{4\}) = 5 + 1 + 2 = 8$.

The SCG model described in [6] can only represent superadditive characteristic functions. In [16] the original definition of SCG was expanded on, by adding the requirement that if a coalition is explicitly listed in W then it cannot be divided further (note: a coalition may now be listed in W even if it gives no synergy):

Example 2: *Consider the characteristic function game $G = \langle N, v\rangle$ where $N = \{1, 2, 3, 4\}$ and the SCG representation W gives the following valuations: $W = \{(\{1\}, 5), (\{2\}, 1), (\{3\}, 1), (\{4\}, 2), (\{1, 2\}, 3)\}$ (i.e. the utility value of agent 1 by itself is 5, while the utility value of the coalition, of agents 1 and 2, is 3)* .

Example values of coalitions that are not explicitly represented are: $v(\{1, 3\}) = v(\{1\}) + v(\{3\}) = 5 + 1 = 6$; and $v(\{\{1, 2, 3\}) = v(\{1, 2\}) + v(\{3\}) = 3 + 1 = 4$ (because $\{1, 2\}$ is in W).

In [16], it was shown that it is guaranteed that there will always be an optimal coalition structure (CS^*) with only coalitions in the SCG. The proof consists of showing that if a coalition C is in a coalition structure and not the SCG, then C can be replaced by a partition of C (consisting of only coalitions in the SCG), which has a value equal to or higher than $v(C)$. This is a result you would expect given Definition 4.

Additionally in [16], it was shown how an CS^* could be identified given the SCG as input. This is a different problem from the one studied in this paper, where the SCG-DP algorithm takes a characteristic function game as input and locates an CS^* while concurrently searching for the SCG.

It should be noted that in this paper the additional requirement of [16] is not used, instead the original [6] definition of a SCG is preferred. We show that the original definition is satisfactory to find an CS^* when it is recognised in the SCG-DP algorithm that the value of a coalition $C \notin W$ is in fact the value of C's highest valued partition. Therefore if C is included in CS^*, then in fact SCG-DP should replace C in CS^* with C's highest value partition, and each coalition in the highest value partition will be replaced if it is not in W (and so on until all the coalitions in CS^* are in W).

3. THEORETICAL FOUNDATIONS

Now that the background has been introduced, the foundations that the SCG-DP algorithm is built upon can be detailed.

In Section 3.1 the properties of a synergy coalition group (SCG) are expanded upon. In Section 3.2 we detail what parts of the search space the SCG-DP algorithm must consider.

3.1 Guaranteeing Stable Solutions

When the characteristic function game is superadditive, then the grand coalition is the optimal coalition structure [18]. In Lemma 2 and Theorem 4 of [6] it is proven that the W set of the SCG allows a core solution to be found if the core is non-empty. This Lemma and Theorem have been extended below to show that a payoff vector in the weak least core can also be guaranteed to be found for the grand coalition using the W set (an example follows the lemma):

LEMMA 3.1. *Given a characteristic function game $\mathcal{G} = \langle N, v\rangle$ and a payoff vector x (where $\sum_{i\in N} x_i = v(N)$), let ϵ^w be the maximum weak ϵ penalty value that still gives an objecting coalition for x given full knowledge on each coalition's value. In this situation, an objecting coalition D for ϵ^w will be present within W.*

PROOF. Suppose x is objected to by a coalition C through $v(C) - |C|\epsilon^w$ because $v(C) - |C|\epsilon^w > x(C)$, and assume that $\forall\epsilon_i^w > \epsilon^w$ there is no objecting coalition. If $(C, v(C)) \in W$ then this proves the lemma. If $(C, v(C)) \notin W$ then from the definition of a SCG, it is known that the value of the coalition can be found through its maximum value partition, i.e. $v(C) - |C|\epsilon^w = \sum_{1\le j\le k}(v(C_j) - |C_j|\epsilon^w)$, for some set of coalitions $P_C = \{C_1, ..., C_k\}$ where:

1. $\bigcup_{j=1}^{k} C_j = C$;
2. $C_i \cap C_j = \emptyset$ for any $i, j \in \{1, ..., k\}$ where $i \neq j$; and
3. $(C_j, v(C_j)) \in W$, for all $C_j \in \{C_1, ..., C_k\}$

Via substitution, it follows that $\sum_{1\le j\le k}(v(C_j) - |C_j|\epsilon^w) > x(C) = \sum_{1\le j\le k} x(C_j)$ and hence for at least one C_j then the following holds: $v(C_j) - |C_j|\epsilon^w > x(C_j)$. Therefore if a coalition C that is not in the SCG representation objects to a payoff vector x given the maximum ϵ^w that still gives an objecting coalition, it has been shown that there exists a coalition $C_j \subset C$, that also objects to x given ϵ^w, yet C_j is represented explicitly within the SCG representation (i.e. $(C_j, v(C_j)) \in W$). Thus the proof is complete. □

Lemma 3.1 shows that given a superadditive characteristic function game, the grand coalition and a payoff vector distributing the grand coalition's value, a coalition that objects to the payoff vector for the maximum weak ϵ penalty value (that allows an objecting coalition) will be present in the SCG representation. For further understanding, consider the following example:

Example 3: *Consider a characteristic function game with $N = \{1, 2, 3, 4\}$ agents, where the coalition $C = \{1, 2, 3\}$ has a value of $v(\{1, 2, 3\}) = 25$. In this example, a payoff vector x gives the coalition C the total payoff of $x(C) = 18$. Therefore it is known that the maximum (integer)* weak ϵ *penalty value for x that C will object to is $\epsilon^w = 2$. This is because $v(C) - |C|\epsilon^w = 25 - (3\times 2) = 25 - 6 = 19 > x(C) = 18$. Coalition $\{1, 2, 3\}$ does* not *object to x when $\epsilon^w = 3$ because $25 - (3 \times 3) = 16 < x(C) = 18$.*

If $C \in W$, then an objecting coalition for the maximum objection-allowing weak ϵ value has been found in the SCG representation. If $C \notin W$ then there must be coalitions within W that make up a partition of C and have an equal or greater combined value than C. For this example assume $(\{1, 2\}, v(\{1, 2\}) = 16), (\{3\}, v(\{3\}) = 9) \in W$. These valuations have been chosen because $v(\{1, 2\}) + v(\{3\}) = v(\{1, 2, 3\})$, i.e. the values total the minimal needed to make sure that coalition $\{1, 2, 3\}$ is not in the SCG representation.

Given theses preliminaries, to stop coalition $\{1, 2\}$ being an objecting coalition when $\epsilon^w = 2$, then $x(\{1, 2\})$ must be greater than or equal to $v(\{1, 2\}) - |\{1, 2\}|\epsilon^w = 16 - (2 \times 2) = 12$. Assume that $x(\{1, 2\}) = 12$ (i.e. the minimal payoff to satisfy the coalition has been given).

Recall that $x(\{1, 2, 3\}) = 18$. Therefore $x_3 = x(\{1, 2, 3\}) - x(\{1, 2\}) = 18 - 12 = 6$. But this gives $v(\{3\}) - |\{3\}|\epsilon^w = 9 - (1 \times 2) = 9 - 2 = 7 > x(\{3\}) = 6$ and so the singleton coalition $\{3\}$ will object to x when $\epsilon^w = 2$ and this objecting coalition has been found in the SCG (as singletons are in W).

In conclusion, this example demonstrates that if an objecting coalition C for the maximum weak ϵ penalty value (that still admits an objecting coalition) is not *present in the SCG representation, then a coalition $C' \subset C$ that is present in the SCG representation will object to the same weak ϵ penalty value.*

The following theorem shows that given the grand coalition, the coalitions in the SCG representation can be used to find a payoff vector within the weak least core:

THEOREM 3.2. *For the grand coalition, a payoff vector that gives the minimum weak ϵ value for stability to occur can be found using only the coalitions in W.*

PROOF. The value of the grand coalition can be allocated by the payoff vector x by solving the following linear program that minimises ϵ:

$$\min \epsilon \text{ subject to:}$$
$$x_i \geq 0 \text{ for each } i \in N \quad (1)$$
$$x(N) = v(N) \quad (2)$$
$$x(C) \geq v(C) - |C|\epsilon, \text{ for all } (C, v(C)) \in W \quad (3)$$

Only the coalitions in W are needed to be used in this linear program, according to Lemma 3.1. □

Before introducing the corollary to this theorem, a formal definition regarding cross-coalition side payments is required, which is shown through the weak ϵ-core$^+$ and weak least core$^+$ definitions:

Definition 5: *The* **weak ϵ-core$^+$***:- For a characteristic function game $\mathcal{G} = \langle N, v\rangle$, a coalition structure and payoff vector pair $\langle CS^*, x\rangle$ is in the weak ϵ-core$^+$ iff:*

$$\sum_{i \in N} x_i = v(CS^*) \qquad \sum_{i \in C} x_i \geq v(C) - |C|\epsilon, \ \forall C \subseteq N$$

The difference of the weak ϵ-core$^+$ compared to the weak ϵ-core of Section 2.2 is that a weak ϵ-core$^+$ payoff vector totals the value of the optimal coalition structure, which may not be the grand coalition. Additionally as the weak ϵ-core$^+$ does *not* have the condition that all the payoff of each coalition in the coalition structure is given to that coalition (i.e. $x(C) = v(C), \forall C \in CS^*$ does *not* have to hold), then cross-coalitional payoff vector transfers can occur. As noted in [1, 11], allowing cross-coalition side payments can benefit multi-agent systems, as it was argued that introducing cross-coalition side payments can be considered a more fair payoff mechanism. The additional fairness comes from eliminating the effect of the coalition structure on agent payoffs. The example given in [1], is that it may be possible that in the optimal coalition structure some agents $M \subset N$ are by themselves in singletons, or are members of a comparatively small coalition compared to the size of the other coalitions in the coalition structure. When cross-coalition side payments are not allowed, these M agents do not benefit from the cooperation of others, even when the M agents are in many potential coalitions that have a high value, which the agents $N \backslash M$ may have used to negotiate for a better payoff. Yet, for the greater good of the population (i.e., maximizing social welfare), the M agents may be forced to stay in the optimal coalition structure.

Given the definition of the weak ϵ-core$^+$, the weak least core$^+$ can be defined:

Definition 6: **weak least core$^+$***:- For a characteristic function game $\mathcal{G} = \langle N, v\rangle$, a coalition structure and payoff vector pair $\langle CS^*, x\rangle$ is in the weak ϵ-core$^+$ iff:*

$$\langle CS^*, x\rangle \text{ is in the weak } \epsilon\text{-core}^+$$
$$\forall \epsilon' < \epsilon, \text{ the weak } \epsilon'\text{-core}^+ \text{ is empty}$$

Given the formal definition regarding cross-coalition side payments, we can introduce the corollary to Theorem 3.2:

COROLLARY 3.2.1. *For the grand coalition of a superadditive cover, a payoff vector in the weak least core$^+$ can be found using only the coalitions in W.*

PROOF. Replace the characteristic function v with the superadditive cover characteristic function v^* in Lemma 3.1 and Theorem 3.2. Now the grand coalition's value will be equal to the optimal coalition structure. This guarantees that a weak least core$^+$ solution for the superadditive cover is found. □

Given Corollary 3.2.1, the SCG-DP algorithm introduced in this paper identifies the following to guarantee that a weak least core$^+$ stable solution is found: (a) the coalitions in the SCG representation W; and (b) an optimal coalition structure that maximises the value of the superadditive cover of the grand coalition. If cross-coalition side payments were not allowed, then it would be much more difficult for the agents to reason over what is the most stable coalition structure and payoff vector pair, as [2, 28] showed that optimal coalition structures may not be the most stable coalition structure. So in this case, multiple coalition structures will have to be compared via not just their value, but also their stability. Searching all possible coalition structures is a highly complex task because the total possible number of coalition structures grows at a significantly higher rate than the number of potential coalitions [27]. For n agents, the number of possible coalition structures is found using the Bell number B_n, which is $\sim \theta(n^n)$ and so significantly larger then the $\theta(2^n)$ growth of possible coalitions [27]. Therefore it is of great benefit to the agents to significantly minimise the number of possible coalition structures to compare.

In this paper, the SCG-DP algorithm assumes cross-coalition side payments are allowed because it significantly reduces the computation costs as multiple coalition structures will not have to be compared according to their stability.

3.2 Search Space Guarantees

Before we detail our search space guarantees, various preliminaries need to be introduced:

Definition 7: *A* ***split*** *of a coalition C, is a two-set partition of C, denoted $[C^x, C^y]$ where:*

$$|C^x| \leq |C^y|, \ C^x \cup C^y = C \ \text{ and } C^x \cap C^y = \emptyset.$$

Additionally the following condition holds (i.e. that if the same size, then C^x has to be smaller lexicographically than C^y):

$$\text{if } |C^x| = |C^y| \text{ then } C^x <_{lex} C^y$$

A split $[C^x, C^y]$ can also be written as $[C^x, -]$, which indicates a split that has only had its first partition generated. The second partition can then be generated subsequently from $N \backslash C^x$ if required.

A split of a coalition can be further divided into *split-partitions*:

Definition 8: *A* ***split-partition*** *is a multi-partition set, denoted $SP = \{C^x_1, ..., C^x_p, C^y_1, ..., C^y_q\}$, and is derived from a two-set partition $[C^x, C^y]$ of an original coalition C. A split-partition has the following properties:*

$$\bigcup_{j=1}^{p} C^x_j = C^x, \qquad \bigcup_{j=1}^{q} C^y_j = C^y,$$
$$C^x_i \cap C^x_j = \emptyset \text{ for any } i, j \in \{1, ..., p\} \text{ where } i \neq j \text{ and}$$
$$C^y_i \cap C^y_j = \emptyset \text{ for any } i, j \in \{1, ..., q\} \text{ where } i \neq j.$$

Therefore the previous definition states that the agents in coalition C^x cannot be mixed with the agents of the coalition C^y in the split partition. Next we define the following functions:

- *AllSplits*(C) returns all the splits that can be made from coalition C.
- *AllSParts*$([C^x, C^y])$ returns all the split-partitions of $[C^x, C^y]$.

Our last preliminary definition introduces the concept of split coverage:

Definition 9: *A split $[C^x, C^y]$ is said to* ***cover*** *a split-partition SP iff $\exists SP' \in$ AllSParts$([C^x, C^y])$ where $\forall C_j \in SP$ then $C_j \in SP'$.*

Example 4: *Consider a coalition $C_1 = \{1, 2, 3, 4, 5, 6\}$ and the splits $[\{2, 3\}, \{1, 4, 5, 6\}]$ and $[\{1\}, \{2, 3, 4, 5, 6\}]$. A split partition of $[\{2, 3\}, \{1, 4, 5, 6\}]$ is $SP_1 = \{\{2, 3\}, \{1\}, \{4\}, \{5, 6\}\}$. Yet SP_1 is* covered *by $[\{1\}, \{2, 3, 4, 5, 6\}]$ because the larger coalition $\{2, 3, 4, 5, 6\}$ can be broken up into $\{2, 3\}$, $\{4\}$ and $\{5, 6\}$. Another split partition of $[\{2, 3\}, \{1, 4, 5, 6\}]$ is $SP_2 = \{\{2, 3\}, \{1, 4\}, \{5, 6\}\}$. But SP_2 is* not covered *by $[\{1\}, \{2, 3, 4, 5, 6\}]$ because, by the definition of a split-partition, $\{1\}$ must be by itself (as $\{1\}$ cannot be grouped with any agents of $\{2, 3, 4, 5, 6\}$).*

As the ODP algorithm showed [14], not all splits for each coalition are needed to be analysed to guarantee that an optimal coalition structure (CS^*) can be found. SCG-DP identifies the SCG of the given characteristic function game, as there is always an CS^* that will only contain coalitions in the SCG [16]. Therefore when SCG-DP searches for CS^* it only considers: (i) coalitions from the SCG that it has identified; and (ii) splits that can lead to an analysis of a new coalition structure involving only SCG coalitions.

When SCG-DP is seeking to identify if a coalition C is in the SCG, it does not have to check every possible split of that coalition. This is because some splits include coalitions not in the SCG and some splits cover only split-partitions previously analysed by another split of C.

The SCG-DP algorithm starts generating splits of a coalition C where the size of the smallest coalition in the split $[C^x, C^y]$ is of size $|C^x| = 1$ and SCG-DP continues generating splits of C until the size of C^x is greater than $\lfloor \frac{|C|}{2} \rfloor$ (i.e. when all possible splits have been generated).

Lemma 3.3 and Theorem 3.4 show that given a split $[C^x, C^y]$ of C where $|C^y| < 2 * |C^x|$, then the value of $[C^x, C^y]$ only needs to be compared to $v(C)$ (to find the best valued partition of C and check if $(C, v(C)) \in W$) if both $(C^x, v(C^x)), (C^y, v(C^y)) \in W$. This is because, when $|C^y| < 2 * |C^x|$, every possible split-partition of $[C^x, C^y]$ will have been covered by a split of C that includes a smaller sized coalition than C^x.

LEMMA 3.3. *Consider a coalition C and a split $[C^x, C^y] \in$ AllSplits(C) where $|C^y| < 2 * |C^x|$. Given $[C^x, C^y]$ and a split-partition $SP \in$ AllSParts$([C^x, C^y])$, then* there exists *another split $[D^x, D^y] \in$ AllSplits(C) where $|D^x| < |C^x|$ and the following holds: $SP \in$ AllSParts$([D^x, D^y])$.*

PROOF. Assume a split $[C^x, C^y] \in$ *AllSplits*(C) has the property $|C^y| < 2 * |C^x|$, and that $SP \in$ *AllSParts*$([C^x, C^y])$ cannot be found by any $[D^x, D^y] \in$ *AllSplits*(C) where $|D^x| < |C^x|$. For this assumption to hold true, then SP cannot contain a partition E^x_i of size $|E^x_i| < |C^x|$, otherwise SP would have been covered by a split that includes D^x where $D^x = E^x_i$.

When $|C^y| < 2 * |C^x|$ then at least one of the partitions in the split-partition SP has a size less than $|C^x|$. If the last coalition E^y_q of the split-partition SP has the size of $|C^x|$ then $\sum_{i=1}^{p} |E^x_i| + \sum_{i=1}^{q-1} |E^y_i| = |C^y| - |C^x|$. But as $|C^x|$ is greater than half of $|C^y|$, then there exists an $E^x_j \leq |C^y| - |C^x| < |C^x|$, which contradicts the assumption as SP will have to contain at least one partition E^x_j with a size smaller than $|C^x|$. □

Exactly how small $|E^x_j|$ is, depends on how many p elements of E^x and q elements of E^y there are in the split-partition.

The following theorem shows that given a split $[C^x, C^y]$ where $|C^y| < 2 * |C^x|$, then this split only needs to be assessed by SCG-DP to find W, if both C^x and C^y are in the SCG:

THEOREM 3.4. *Consider a coalition C and a split $[C^x, C^y] \in$ AllSplits(C) where $|C^y| < 2 * |C^x|$. Then the SCG-DP algorithm needs to compare $v(C)$ to $f_2[C^x] + f_2[C^y]$ as part of the checks to see if $(C, v(C)) \in W$, only if $(C^x, v(C^x)), (C^y, v(C^y)) \in W$.*

PROOF. According to Lemma 3.3, each split-partition $SP \in$ *AllSParts*$([C^x, C^y])$ has been covered by a split $[D^x, D^y]$ where $|D^x| < |C^x|$. Therefore the value of each SP has already been compared against $v(C)$ when analysing $[D^x, D^y]$ because SCG-DP generates the splits according to the size of the smallest coalition in it (from size 1 to $\lfloor \frac{|C|}{2} \rfloor$). Therefore, only the split $[C^x, C^y]$ (i.e. the split itself and not any split-partitions) has not been compared against $v(C)$. But if one of C^x or C^y is not in W, then one of $[C^x, C^y]$'s split-partitions were of greater or equal value than $[C^x, C^y]$ and so the following is guaranteed to hold: $v(C) \geq f_2(C^x) + f_2(C^y)$. □

The next lemma and theorem show that given a split $[C^x, C^y]$ of C where $|C^y| \geq 2 * |C^x|$, then the value of $[C^x, C^y]$ needs to be compared to $v(C)$ if C^x in the SCG. This is because, when $|C^y| \geq 2 * |C^x|$, there are many possible split-partitions of $[C^x, C^y]$ that will *not* have been covered by a split with a smaller sized coalition than C^x.

LEMMA 3.5. *Consider a coalition C and a split $[C^x, C^y] \in$ AllSplits(C) where $|C^y| \geq 2 * |C^x|$. Given $[C^x, C^y]$ and a split-partition $SP \in$ AllSParts$([C^x, C^y])$, then* there may not exist *a split $[D^x, D^y] \in$ AllSplits(C) where $|D^x| < |C^x|$ and $SP \in$ AllSParts$([D^x, D^y])$.*

PROOF. Assume a split $[C^x, C^y] \in$ *AllSplits*(C) has the property $|C^y| \geq 2 * |C^x|$, and that $SP \in$ *AllSParts*$([C^x, C^y])$ can always be found by any $[D^x, D^y] \in$ *AllSplits*(C) where $|D^x| < |C^x|$. For this assumption to hold true, then SP cannot contain only partitions of sizes $\geq |C^x|$, otherwise SP would *not* have been covered by a split that included D^x.

When $|C^y| \geq 2 * |C^x|$ then at least one split-partition SP can be found that includes only partitions that have sizes $\geq |C^x|$. For example, a 3-part split-partition of sizes $|C^x|$, $\lfloor \frac{|C^y|}{2} \rfloor$ and $\lceil \frac{|C^y|}{2} \rceil$ can be made, and because $|C^y| \geq 2 * |C^x|$ the assumption is contradicted. □

The following theorem shows that given a split $[C^x, C^y]$ where $|C^y| \geq 2 * |C^x|$, then this split only needs to be assessed by SCG-DP to find W, if C^x is in the SCG:

THEOREM 3.6. *Consider a coalition C and a split $[C^x, C^y] \in$ AllSplits(C) where $|C^y| \geq 2 * |C^x|$. Then the SCG-DP algorithm needs to compare $v(C)$ to $f_2[C^x] + f_2[C^y]$ as part of the checks to see if $(C, v(C)) \in W$, only if $(C^x, v(C^x)) \in W$.*

PROOF. According to the previous lemma, a split-partition $SP \in$ *AllSParts*$([C^x, C^y])$ may not have been covered by a split $[D^x, D^y]$ when: (a) SP only includes partitions of sizes greater than or equal to $|C^x|$; (b) $|D^x| < |C^x|$; and (c) $|C^y| \geq 2 * |C^x|$. All split-partitions that include partitions of sizes greater or equal to $|C^x|$,

Algorithm 1: The `SCG-DP` algorithm to find an optimal coalition structure and a stable payoff vector.

1: **function** `SCG-DP(`N`,` v`)`
2: **Input:** $\langle N, v\rangle$; where N is the global set of agent IDs and v is the characteristic function.
3: **Output:** $\langle CS^*, x\rangle$; where CS^* is the optimal coalition structure and x is a stable payoff vector.
4: **begin**;
5: $W = \emptyset$; // holds the SCG representation
6: **for** $i \in N$ **do**
7: *set* $f_1[\{i\}] := \{i\}$, $f_2[\{i\}] := v(\{i\})$ and $(\{i\}, f_2[\{i\}]) \in W$;
8: **end for**
9: **for** $s := 2$ to n **do**
10: **for** each $C \subseteq N$ where $|C| = s$ **do**
11: $\Psi =$ `EvalSplits`(C, W)
12: $f_2[C] :=$ $\max\{v(C), \max_{[C^x, C^y] \in \Psi}(f_2[C^x] + f_2[C^y])\}$;
13: **if** $f_2[C] < v(C)$ **then**
14: *set* $f_1[C] := C$ and $(C, f_2[C]) \in W$;
15: **else**
16: *set* $f_1[C] := C^*$ where C^* maximises $f_2[C]$;
17: **end if**
18: **end for**
19: **end for**
20: *set* $CS^* := \{N\}$
21: **for** each $C_i \in CS^* = \{C_1, ...C_k\}$ **do**
22: **if** $C_i \notin W$ **then**
23: *set* $CS^* := (CS^* \backslash \{C\}) \cup \{f_1[C]\}$;
24: *restart* this for loop from C_i;
25: **end if**
26: **end for**
27: $x =$ `FindStablePayoff(`W`)`
28: **return** $\langle CS^*, x\rangle$;
29: **end**;

must include C^x due to the definition of split-partitions. Yet if $(C^x, v(C^x)) \notin W$ then this split does not need to be analysed anyway as an optimal coalition structure can always be found using only the coalitions in W. □

To conclude this section, the SCG-DP algorithm implements implements Theorem 3.4 and Theorem 3.6 (to make sure that the necessary splits are analysed) via its `EvalSplits` function (see Algorithm 2) in lines 8 to 11 and lines 12 to 14 respectively. In the `EvalSplits` function, C^y is generated only when needed, to save on computation, therefore both of the following are used in the function: (a) $|C^y|$ is substituted with $n - |C^x|$; and (b) C^y is only generated if $(C^x, v(C^x)) \in W$.

4. THE SCG-DP ALGORITHM

The SCG-DP algorithm (see Algorithm 1) takes a characteristic function game as input and follows the style of the other dynamic programming algorithms in the literature. That is, SCG-DP breaks the coalition structure generation problem up into finding the highest valued partition of each subset of the grand coalition in size order from smallest subset to largest (line 6 and 9).

To find an optimal coalition structure (CS^*), like DP of [34], the SCG-DP algorithm uses: (a) the f_1 table where $f_1[C]$ is set to equal the highest valued split of C if this is greater or equal to $v(C)$, otherwise $f_1[C] = C$ (lines 14 and 16); and (b) the f_2 table

Algorithm 2: The `EvalSplits` function finds which splits of the coalition C should be checked by `SCG-DP` to find out whether C is in the SCG representation.

1: **function** `EvalSplits(`C`,` W`)`
2: **Input:** $\langle C, W\rangle$; where C is the coalition to assess the splits of, and W is the SCG representation.
3: **Output:** $\Psi = \{[C^x, C^y]_1, ..., [C^x, C^y]_k\}$;
4: **begin**;
5: $\Psi = \emptyset$; $p = 1$;
6: **for** $r = 1$ to $r > \lfloor \frac{|C|}{2} \rfloor$ **do**
7: **for** each $[C^x, -] \in AllSplits(C)$ where $|C^x| = r$ **do**
8: **if** $n - |C^x| < 2 * |C^x|$ **and** $(C^x, f_2[C^x]) \in W$ **then**
9: **if** $(C^y, f_2[C^y]) \in W$, where $C^y = N \backslash C^x$ **then**
10: $[C^x, C^y]_p \in \Psi$; $p++$;
11: **end if**
12: **else if** $n - |C^x| \geq 2 * |C^x|$ **and** $(C^x, f_2[C^x]) \in W$ **then**
13: $[C^x, C^y]_p \in \Psi$, where $C^y = N \backslash C^x$; $p++$;
14: **end if**
15: **end for**
16: **end for**
17: **return** Ψ;
18: **end**;

where $f_2[C]$ holds the value of C (if $f_1[C] = C$) or the value of C's highest-valued split (line 12). But crucially the difference between SCG-DP and DP is that SCG-DP's search for the CS^* does not involve considering every possible split of every coalition $C \subseteq N$. Only the splits returned by the `EvalSplits` function are considered (line 11). The ODP algorithm of [14] also does not consider every possible split when searching for an CS^*. The SCG-DP and ODP algorithms are compared in Section 5.

Additionally, SCG-DP, unlike the DP and ODP algorithms, collects the SCG representation in W at two points: (1) Firstly by the definition of the SCG, the singleton coalitions should be in W (line 7); and (2) When every split-partition of C has a value less than $v(C)$ (line 13 and 14). In practice, if C is in W can be recorded simply via a boolean, as the value of C is saved in f_2.

After f_1 and f_2 are computed, the optimal coalition structure CS^* is found recursively in the same manner as the DP algorithm. This is performed by first setting the CS^* equal to the grand coalition (line 20) and then performing a loop (lines 21 to 26) where all the coalitions in the CS^* are checked to see if they are in the SCG (line 22). If a coalition $C \notin W$, then it needs to be replaced by its highest valued partition. This process continues until all the coalitions in CS^* are members of the SCG.

Finally, the coalitions that have been collected into W are used to find a stable payoff vector (line 27). Details of this are described in Section 4.1. Once the stable payoff vector has been found, the characteristic function game outcome will be returned (line 28).

4.1 Solution Concept Satisfiability

As has been described, the SCG-DP algorithm locates all the coalitions in the SCG representation (i.e. it finds W). Given W, various guarantees can be made on the payoff vector x that is returned from the `FindStablePayoff` function (which follows Corollary 3.2.1 in Section 3.1). These guarantees are provided via Theorem 3.2 and Corollary 3.2.1 and depend on the characteristic function game given as input to the SCG-DP algorithm:

- A **core** solution will be found when the characteristic function game has a superadditive v.

- A **weak-least core** solution will be found when the grand coalition is the optimal coalition structure but not core-stable.
- A **weak-least core**$^+$ solution will be found when the grand coalition is *not* the optimal coalition structure. In this case, cross-coalition side payments may occur.

The superadditive cover characteristic function v^*, referred to in Corollary 3.2.1, has been found in the SCG-DP via the f_2 table.

If cross coalition side payments are undesirable, SCG-DP could be modified to find the most stable CS^*. The section of the SCG-DP that finds an CS^* can instead find all the CS^*s. Then each CS^* can be compared via stability. This procedure is known as the optimality gap [28] and significantly cuts down on the number of coalition structures to compare via their stability, as the number of CS^*s is usually significantly less than all the coalition structures.

4.2 An Example of the Algorithm

To show how SCG-DP works, due to space considerations, we focus on the evaluation of two coalitions of size $s = 4$ in a $n = 5$ agent characteristic function game. The following evaluation is also presented in Table 1. As the coalitions are evaluated in size order (see line 6 and 9 of SCG-DP), then coalitions of size $1, 2$ and 3 have already been analysed by SCG-DP. Within those sizes, the following were found to be in the SCG (in this example): $(\{1\}, 1), (\{2\}, 1.2), (\{3\}, 0.8), (\{4\}, 1.2), (\{5\}, 0.9), (\{1, 3\}, 2)$, $(\{2, 5\}, 2.3), (\{3, 5\}, 2), (\{2, 3, 5\}, 3.7) \in W$.

Now for a coalition of size 4, there are 7 different splits, which are shown in the first 7 lines of the second column of Table 1 for each coalition. To find out if a split should be compared to the coalition's value (displayed in the 8th line of column 2 for each coalition), the `EvalSplit` function generates the first coalition C^x of the split and generates the second coalition C^y if $C^x \in W$. Then if the split $[C^x, C^y]$'s value is compared to the value of the coalition depends on the size of C^x, as shall be explained.

Take the first split of coalition $\{1, 2, 3, 4\}$, i.e. $[\{1\}, \{2, 3, 4\}]$. This split satisfies the following: $n - |C^x| = 5 - 1 \geq 2 * |C^x|$ (where $C^x = \{1\}$). Therefore lines 12 to 14 of the `EvalSplit` function is run. As singletons are always in the SCG, then line 13 states that C^y should be generated (hence coalition $\{2, 3, 4\}$ is shown in the table) and that this split should be analysed (hence a numeric result is shown in the table). The value 4.2 is found as $v(\{1\}) = 1$ and the highest valued disjoint partition of $\{2, 3, 4\}$ in the SCG has a value of 3.2, because $v(\{2\}) + v(\{3\}) + v(\{4\}) = 1.2 + 0.8 + 1.2 = 3.2$. Alternatively, in the second line, the best valued partition of $\{1, 3, 4\}$ is also 3.2 but this time because $v(\{1, 3\}) + v(\{4\}) = 2 + 1.2 = 3.2$.

Now consider splits of the form $[C^x, C^y]$ where $|C^x| = |C^y| = 2$ for the same coalition $\{1, 2, 3, 4\}$. The first split of this kind is $[\{1, 2\}, \{3, 4\}]$. This split satisfies the following: $n - |C^x| = 5 - 2 < 2 * |C^x|$ (where $C^x = \{1, 2\}$). So lines 8 to 11 of the `EvalSplit` function are run. As $\{1, 2\} \notin W$, the if statement of line 8 evaluates to false, therefore the C^y coalition of the split is not generated (hence $-$ is shown in the table) and the split is not evaluated (hence N/A is shown in the table). Alternatively, for the split $[\{1, 3\}, \{2, 4\}]$, $C^y = \{2, 4\}$ is generated because $\{1, 3\} \in W$ but the split is not evaluated as $\{2, 4\} \notin W$.

5. EXPERIMENTAL EVALUATION

This section will provide an evaluation through the discussion of: (a) the number of split operations to be performed for different agent numbers; and (b) the number of coalition lookup operations to be performed for different agent numbers.

Coalition	The evaluations performed before setting f_1 and f_2	f_1	f_2	In W?
$\{1, 2, 3, 4\}$	$f_2[\{1\}] + f_2[\{2, 3, 4\}] = 4.2$ $f_2[\{2\}] + f_2[\{1, 3, 4\}] = 4.4$ $f_2[\{3\}] + f_2[\{1, 2, 4\}] = 4.2$ $f_2[\{4\}] + f_2[\{1, 2, 3\}] = 4.4$ $f_2[\{1, 2\}] + f_2[-] = \text{N/A}$ $f_2[\{1, 3\}] + f_2[\{2, 4\}] = \text{N/A}$ $f_2[\{1, 4\}] + f_2[-] = \text{N/A}$ $v[\{1, 2, 3, 4\}] = 4.6$	$\{1, 2, 3, 4\}$	4.6	Yes
$\{1, 2, 3, 5\}$	$f_2[\{1\}] + f_2[\{2, 3, 5\}] = 4.7$ $f_2[\{2\}] + f_2[\{1, 3, 5\}] = 4.1$ $f_2[\{3\}] + f_2[\{1, 2, 5\}] = 4.1$ $f_2[\{5\}] + f_2[\{1, 2, 3\}] = 4.1$ $f_2[\{1, 2\}] + f_2[-] = \text{N/A}$ $f_2[\{1, 3\}] + f_2[\{2, 5\}] = 4.3$ $f_2[\{1, 5\}] + f_2[-] = \text{N/A}$ $v[\{1, 2, 3, 5\}] = 4.3$	$\{1\}\{2, 3, 5\}$	4.7	No

Table 1: This table details the splits and coalitions generated, for coalitions $\{1, 2, 3, 4\}$ and $\{1, 2, 3, 5\}$ in the characteristic function game described in Section 4.2.

An experimental analysis was used as both (a) and (b) are dependent on the coalition-value distribution, which the SCG-DP algorithm has no *a priori* knowledge on. The experiments were run using: (i) different agent numbers from 5 to 20, where 30 runs for each agent number were used; and (ii) differing coalition-value distributions, which include normal, uniform and NDCS, as is usual in the coalition structure generation literature (see [12, 15, 23, 25]). These coalition-value distributions are defined as:

- *Uniform*:- Each coalition's value is determined by multiplying the number of agents in the coalition by a variable q picked from a uniform distribution between 0 and 1.0. Formally, $v(C) = max(0, |C| \times q)$, where: $q \in \mathcal{U}(a, b)$; $a = 0$; and $b = 1.0$.
- *Normal*:- Each coalition's value is determined by multiplying the number of agents in the coalition by a variable q picked from a normal distribution with mean 1 and variance 0.1. Formally, $v(C) = max(0, |C| \times q)$, where: $q \in \mathcal{N}(\mu, \sigma^2)$; $\mu = 1$; and $\sigma^2 = 0.1$.
- *NDCS*:- Each coalition's value is determined by a variable q picked from a normal distribution with mean of the coalition's size and variance also according to the coalition's size. Formally, $v(C) = max(0, q)$, where: $q \in \mathcal{N}(\mu, \sigma^2)$; $\mu = |C|$; and $\sigma^2 = |C|$.
- *Subadditive*:- A characteristic function game is subadditive if for any two disjoint coalitions D and E, where $D, E \subset N$ and $D \cap E = \emptyset$, then $v(D \cup E) \leq v(D) + v(E)$.
- *Strictly superadditive*:- A characteristic function game is strictly superadditive if for any two disjoint coalitions D and E, where $D, E \subset N$ and $D \cap E = \emptyset$, then $v(D \cup E) > v(D) + v(E)$.

In this section the SCG-DP algorithm is compared to the Optimal Dynamic Programming (ODP) algorithm described in [14] for the above coalition-value distributions. SCG-DP seeks to minimise the number of split operations to find an optimal and stable coalition structure (whatever the coalition-value distribution), while ODP minimises the number of split operations to find the optimal coalition structure (whatever the coalition-value distribution).

5.1 Number of Analysed Splits

The number of analysed splits for a coalition C is defined as the splits of C that are considered for $f_2[C]$. In SCG-DP, this equals

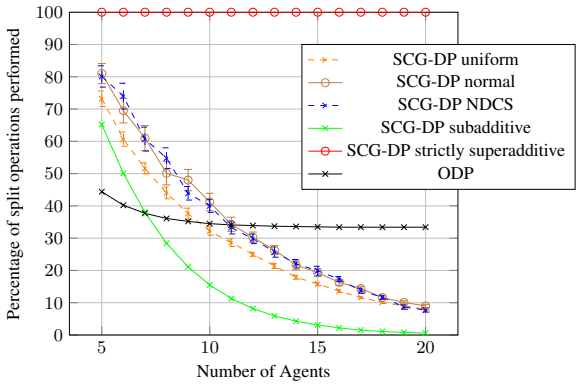

Figure 1: This figure displays the percentage of split operations performed in SCG-DP (for different coalition-value distributions) and in ODP (for any coalition-value distribution), compared to the total possible split operations of the DP algorithm, which is equivalent to the Stirling number of the second kind: $S(n+1,3)$. (95% confidence interval displayed).

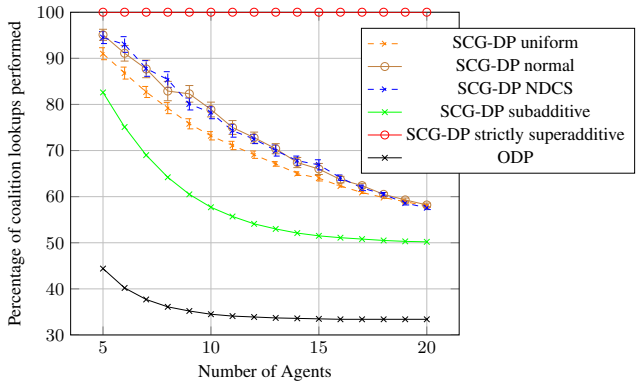

Figure 2: This figure displays the percentage of coalition look up operations performed in SCG-DP (for different coalition-value distributions) and ODP (for any coalition-value distribution) compared to the total possible lookup operations of the DP algorithm, which is equivalent to two times the Stirling number of the second kind: $2 \times S(n+1,3)$. (95% confidence interval displayed).

the number of splits returned by the `EvalSplits` function. The total number of analysed splits is the number of analysed splits for every coalition $C \subseteq N$.

The total number of analysed splits results for SCG-DP for five different coalition-value distributions are displayed in Figure 1. The SCG-DP results are taken from the experiments discussed earlier in this section. Additionally in Figure 1 the ODP algorithm results are detailed, which are found in [14].

As can be seen in Figure 1, as n increases SCG-DP analyses fewer splits than ODP for all non-strictly superadditive coalition-value distributions. This is due to the additional pre-processing `EvalSplits` function of SCG-DP. As ODP does not have a similar pre-processing stage, a fairer comparison method would be to compare via the number of coalition lookups, as discussed next.

5.2 Number of Coalition Lookups

A coalition lookup is defined as the generation of a coalition within a split (and therefore the lookup of the coalition's information). In ODP, as the number of analysed splits does not depend on the coalition-value distribution, then the number of coalition lookups also does not depend on the coalition-value distribution.

For SCG-DP, coalition lookups occur in the `EvalSplits` function. Coalition C^x is generated at line 7 for every split. Then coalition C^y is generated if C^x is in the SCG, at line 9 or line 13. It is then assumed that the information of the looked up coalitions stays in memory until the next time the `EvalSplits` function is run (so that the coalitions do not have to be looked up again). As the generation of C^y is dependant on C^x being in the SCG, the number of coalition lookups is variable in SCG-DP.

The total number of coalition look ups for SCG-DP for the five different coalition-value distributions are displayed in Figure 2. Additionally in Figure 2, the ODP results are detailed.

For SCG-DP, the subadditive experiments have the minimal number of lookups and act as a lower bound. This lower bound occurs because subadditive games have the minimal number of coalitions in the SCG; the n singleton coalitions. As can be seen, the number of lookups in the subadditive experiments are converging onto 50%. This is because C^x always has to be generated in the `EvalSplits` function, and C^y only has to be generated when $|C^x| = 1$ (i.e. when C^x is a singleton coalition). The number of times C^y gets generated as n increases becomes more and more insignificant as a percentage of the total coalition lookups.

Conversely, the strictly superadditive coalition-value distribution acts as an upper bound for SCG-DP. This upper bound occurs because strictly superadditive games have the maximum number of coalitions in the SCG; all the coalitions. In this case SCG-DP performs the same number of lookups as the original DP algorithm [34] (i.e. 100%). That said, for the other three more common coalition-value distributions for coalition structure generation algorithms (uniform, normal and NDCS) [12, 23], the number of coalition lookups is tending towards the lower bound as n is increasing.

6. CONCLUSION AND FUTURE WORK

In this paper, the *Synergy Coalition Group-based Dynamic Programming* (SCG-DP) algorithm was presented, which takes any characteristic function game as input and finds an optimal coalition structure as well as a stable payoff vector by identifying all the coalitions in the Synergy Coalition Group (SCG). SCG-DP locates the SCG without checking the entire search space identified by the Dynamic Programming (DP) algorithm of [34]. The optimal dynamic programming algorithm of [14] showed that only approximately 33% of the search operations of DP are required to guarantee that an optimal coalition structure is found. In this paper, our experimental results show that the lower bound, to guarantee that an optimal coalition structure and a weak least core stable payoff vector is found, converges onto 50% as the number of agents increases. The upper bound remains at 100%, but for standard coalition-value distributions in the literature, the results are trending towards the lower bound as the number of agents increases.

Two possible strands of future work are as follows: (1) SCG-DP only gives a solution upon completion, so exploring an any-time version of SCG-DP could be beneficial; and (2) SCG-DP is a centralised algorithm, yet decentralised methods for the first stage of coalition formation - performing the coalition value calculations (CVC) - exist (e.g. [20, 26, 31, 32]). Building on one of the decentralised CVC algorithms to create a decentralised version of SCG-DP (in a similar manner to how [15] built on [20]), can help spread the computational burden while also making SCG-DP more robust.

REFERENCES

[1] S. Airiau and S. Sen. On the stability of an optimal coalition structure. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, pages 203–308, 2010.

[2] S. Brânzei and K. Larson. Coalitional affinity games and the stability gap. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 79–84, 2009.

[3] J. C. Cesco. A convergent transfer scheme to the core of a TU-game. *Revista de Matematicas Aplicadas*, 19:23–35, 1998.

[4] A. Chapman, R. A. Micillo, R. Kota, and N. R. Jennings. Decentralised dynamic task allocation using overlapping potential games. *The Computer Journal*, 53:1462–1477, 2010.

[5] K. Chatterjee, B. Dutta, and K. Sengupta. A noncooperative theory of coalitional bargaining. *Review of Economic Studies*, 60:463–477, 1993.

[6] V. Conitzer and T. W. Sandholm. Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, 170:607–619, 2006.

[7] V. D. Dang and N. R. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the 3th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 564–571, 2004.

[8] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani. *Algorithms*. McGraw-Hill Higher Education, 2006.

[9] R. Evans. Coalitional bargaining with competition to make offers. *Games and Economic Behaviour*, 19:211–220, 1997.

[10] D. Gillies. *Some theorems on n-person games*. PhD thesis, Princeton University, 1953.

[11] A. Iwasaki, S. Ueda, and M. Yokoo. Finding the core for coalition structure utilizing dual solution. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*, pages 114–121, 2013.

[12] K. S. Larson and T. W. Sandholm. Anytime coalition structure generation: An average case study. *Journal of Experimental and Theoretical AI*, 12:23–42, 2000.

[13] E. Lehrer. Allocation processes in cooperative games. *International Journal of Game Theory*, 31:341–351, 2003.

[14] T. Michalak, T. Rahwan, E. Elkind, M. Wooldridge, and N. R. Jennings. A hybrid exact algorithm for complete set partitioning. *Artificial Intelligence*, 230:14 – 50, 2016.

[15] T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. McBurney, and N. R. Jennings. A distributed algorithm for anytime coalition structure generation. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent System (AAMAS)*, pages 1007–1014, 2010.

[16] N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, and M. Yokoo. Coalition structure generation utilizing compact characteristic function representations. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 623–638, 2009.

[17] A. Okada. A noncooperative coalitional bargaining game with random proposers. *Games and Economic Behaviour*, 16:97–108, 1996.

[18] M. J. Osbourne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[19] K. Pawlowski, K. Kurach, K. Svensson, S. D. Ramchurn, T. Michalak, and T. Rahwan. Coalition structure generation with the graphics processing unit. In *Proceedings of the 13th International conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 293–300, 2014.

[20] T. Rahwan and N. R. Jennings. An algorithm for distributing coalition value calculations among cooperating agents. *Artificial Intelligence*, 171:535–567, 2007.

[21] T. Rahwan and N. R. Jennings. Coalition structure generation: Dynamic programming meets anytime optimization. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, pages 156–161, 2008.

[22] T. Rahwan and N. R. Jennings. An improved dynamic programming algorithm for coalition structure generation. In *Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1417–1420, 2008.

[23] T. Rahwan, T. Michalak, and N. R. Jennings. A hybrid algorithm for coalition structure generation. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI)*, pages 1443–1449, 2012.

[24] T. Rahwan, T. Michalak, M. Wooldridge, and N. R. Jennings. Coalition structure generation: A survey. *Artificial Intelligence*, 229:139 – 174, 2015.

[25] T. Rahwan, S. D. Ramchurn, A. Giovannucci, and N. R. Jenning. An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, 34:521–567, 2009.

[26] L. Riley, K. Atkinson, P. Dunne, and T. R. Payne. Distributing coalition value calculations to coalition members. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*, pages 2117–2123, 2015.

[27] T. W. Sandholm, K. S. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111:209–238, 1999.

[28] T. C. Service. *Coalition Structure Generation In Characteristic Function Games*. PhD thesis, Vanderbilt University, 2012.

[29] T. C. Service and J. A. Adams. Constant factor approximation algorithms for coalition structure generation. *Autonomous Agents and Multi-Agent Systems*, 23(1):1–17, 2011.

[30] L. S. Shapley and M. Shubik. Quasi-cores in a monetary economy with non-convex preferences. *Econometrica*, 34:805–827, 1966.

[31] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101:165–200, 1998.

[32] M. Vinyals, F. Bistaffa, A. Farinelli, and A. Rogers. Coalitional energy purchasing in the smart grid. In *Proceedings of the IEEE International Energy Conference & Exhibition (ENERGYCON)*, pages 848–853, 2012.

[33] L. S.-Y. Wu. A dynamic theory for the class of games with nonempty cores. *SIAM Journal on Applied Mathematics*, 32:328–338, 1977.

[34] D. Y. Yeh. A dynamic programming approach to the complete set partitioning problem. *BIT Numerical Mathematics*, 26:467–474, 1986.